**Tic Tac Toe Game- Code Documentation**

The provided code is a simple implementation of a Tic Tac Toe game using the Streamlit library in Python. The game can be played by two players, either human or computer. Below is a detailed documentation of the code.

**1. Initialization:**

The `init` function is responsible for initializing the game state. It sets up the game board, player turn, and other necessary variables. If the `post\_init` parameter is `True`, it resets the game for a new match.

| def init(post\_init=False):  """  Initializes the game state.   Parameters:  - post\_init (bool): If True, resets the game for a new match.  """  if not post\_init:  st.session\_state.opponent = 'Human'  st.session\_state.board = np.full((3, 3), '⚫', dtype=str)  st.session\_state.player = '❌'  st.session\_state.warning = False   st.session\_state.winner = None  st.session\_state.over = False |
| --- |

**2. Styling:**

The game interface is styled using a custom CSS style defined in the `main\_styles` variable. It sets a background image and text color for the Streamlit app.

| main\_styles = """  <style>  .stApp {  background-image:url("https://img.freepik.com/free-photo/empty-blackboard\_53876-30426.jpg");  color: white;  background-size:100%  }  </style> """ # Display custom styles st.markdown(main\_styles, unsafe\_allow\_html=True) |
| --- |

**3. Move Availability:**

The `check\_available\_moves` function returns a list of available moves on the current game board. The optional parameter `extra` determines whether to return the list of moves as indices or as coordinates.

| def check\_available\_moves(extra=False) -> list:  """  Checks and returns the available moves on the current game board.   Parameters:  - extra (bool): If True, returns the moves as coordinates; otherwise, returns indices.   Returns:  - list: List of available moves.  """  raw\_moves = [row for col in st.session\_state.board.tolist() for row in col]  num\_moves = [i for i, spot in enumerate(raw\_moves) if spot == '⚫']  if extra:  return [(i // 3, i % 3) for i in num\_moves]  return num\_moves |
| --- |

**4. Winning Conditions:**

The `check\_rows` and `check\_diagonals` functions check if there is a winner in the current game state by examining rows and diagonals, respectively.

| def check\_rows(board):  """  Checks for a winner in the rows.   Parameters:  - board: The game board.   Returns:  - str or None: The winning player or None.  """  for row in board:  if len(set(row)) == 1:  return row[0]  return None  def check\_diagonals(board):  """  Checks for a winner in the diagonals.   Parameters:  - board: The game board.   Returns:  - str or None: The winning player or None.  """  if len(set([board[i][i] for i in range(len(board))])) == 1:  return board[0][0]  if len(set([board[i][len(board) - i - 1] for i in range(len(board))])) == 1:  return board[0][len(board) - 1]  return None |
| --- |

**5. State Checking:**

The `check\_state` function checks and displays the game state. It shows a success message if a player has won, a warning if a move is invalid, and an info message if the game is a tie.

| def check\_state():  """  Checks and displays the game state.  """  if st.session\_state.winner:  st.success(f"Congrats! {st.session\_state.winner} won the game! 🎈")  st.balloons()  if st.session\_state.warning and not st.session\_state.over:  st.warning('⚠️ This move already exists')  if st.session\_state.winner and not st.session\_state.over:  st.session\_state.over = True  st.session\_state.win[st.session\_state.winner] = (  st.session\_state.win.get(st.session\_state.winner, 0) + 1  )  elif not check\_available\_moves() and not st.session\_state.winner:  st.info(f'It\'s a tie 📍')  st.session\_state.over = True |
| --- |

| **def check\_win(board):  """  Checks for a winner in the game.   Parameters:  - board: The game board.   Returns:  - str or None: The winning player or None.  """  for new\_board in [board, np.transpose(board)]:  result = check\_rows(new\_board)  if result:  return result  return check\_diagonals(board)** |
| --- |

**6. Computer Player**:

The `computer\_player` function is called when the computer is the opponent. It randomly selects an available move and updates the game state.

| def computer\_player():  """  Handles the computer player's move.  """  moves = check\_available\_moves(extra=True)  if moves:  i, j = random.choice(moves)  handle\_click(i, j) |
| --- |

**7. Handling Clicks:**

The `handle\_click` function is called when a player makes a move. It checks if the move is valid, updates the board, changes the player turn, and checks for a winner.

| def handle\_click(i, j):  """  Handles a player's move.   Parameters:  - i (int): Row index.  - j (int): Column index. """  if (i, j) not in check\_available\_moves(extra=True):  st.session\_state.warning = True  elif not st.session\_state.winner:  st.session\_state.warning = False  st.session\_state.board[i, j] = st.session\_state.player  st.session\_state.player = "🟢" if st.session\_state.player == "❌" else "❌"  winner = check\_win(st.session\_state.board)  if winner != "⚫":  st.session\_state.winner = winner |
| --- |

**8. Main Game Logic:**

The `main` function is the main logic of the game. It displays the game title, initializes the game state, and provides a sidebar for options like starting a new game or choosing the opponent (Human or Computer). The game board is displayed using Streamlit columns, and player moves are handled by clicking buttons.

| def main():  """  Main function for the Tic Tac Toe game.  """  st.write(  """  # ❌🟢 Tic Tac Toe 🟢❌  """  )   if "board" not in st.session\_state:  init()   # Sidebar section  with st.sidebar:  st.image('https://www.damemagazine.com/wp-content/uploads/2021/06/mother-lesbian-DSM.jpg')  st.write(" ")  st.write(" ")  st.button('New game', on\_click=init, args=(True,))  st.radio('Set opponent',  ['Human', 'Computer'],  key='opponent',  on\_change=init,  args=(True,))  st.write(" ")  st.write(" ")   # Main section  st.button(  f'{"❌" if st.session\_state.player == "❌" else "⭕"}\'s turn'  if not st.session\_state.winner  else f'🏁 Game finished'  )  with st.container():  for i, row in enumerate(st.session\_state.board):  cols = st.columns([5, 1, 1, 1, 5])  for j, field in enumerate(row):  cols[j + 1].button(  field,  key=f"{i}-{j}",  on\_click=handle\_click  if st.session\_state.player == '❌'  or st.session\_state.opponent == 'Human'  else computer\_player(),  args=(i, j),  )   check\_state() |
| --- |

**9. Running the Game:**

The `\_\_main\_\_` block ensures that the `main` function is executed when the script is run.

**10. Interface Elements:**

- The game title is displayed using a Streamlit markdown.

- The opponent can be selected as either a human or a computer in the sidebar.

- Buttons are used for player moves, and the current player's turn is displayed.

- The game state, including warnings, winner messages, and ties, is displayed using Streamlit components.

Overall, the code provides a simple and interactive implementation of Tic Tac Toe with a clean interface, making it easy for users to play against each other or against the computer.